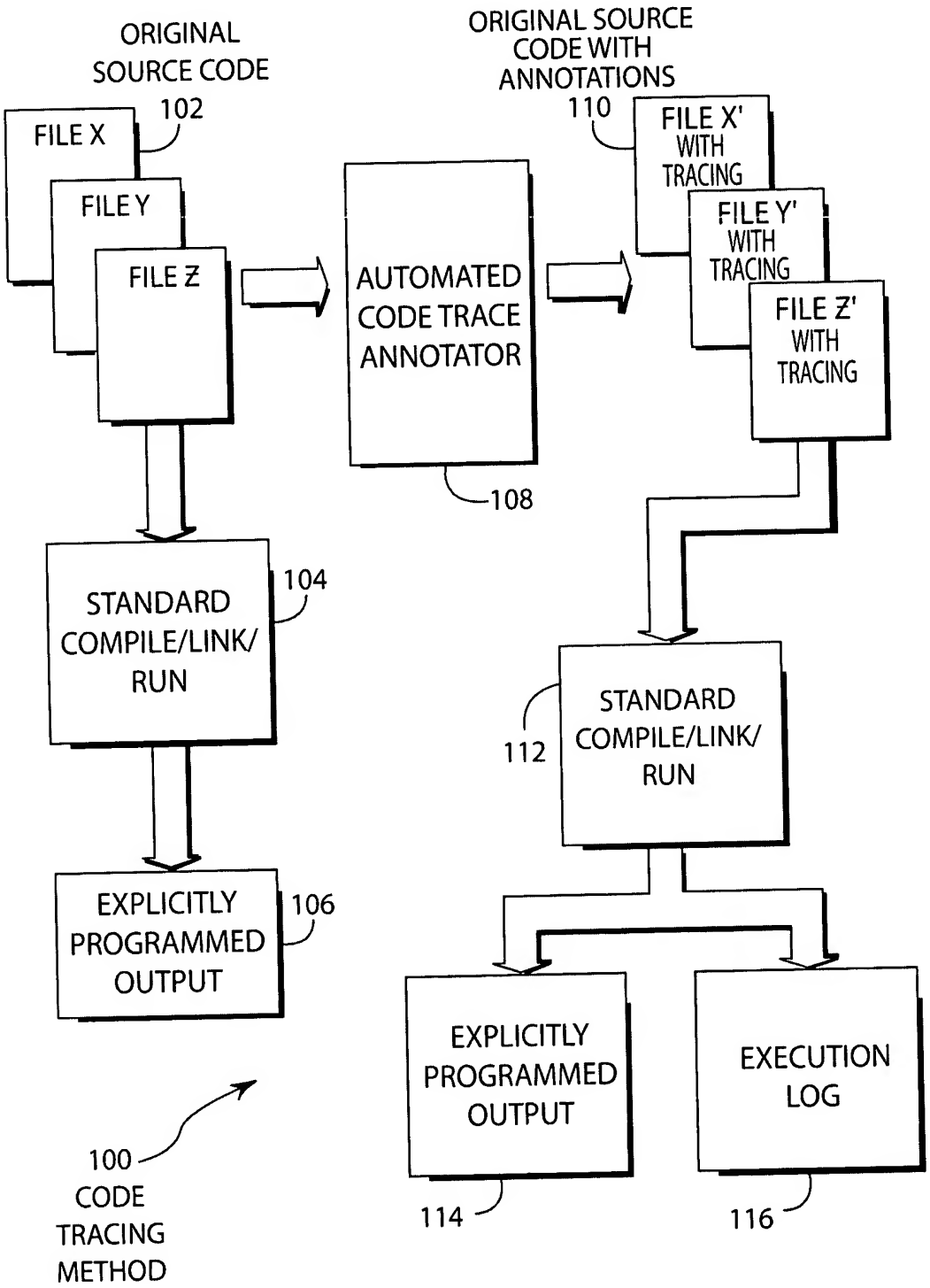

ORIGINAL SOURCE CODE
102
FILE X
FILE Y
FILE Z
AUTOMATED CODE TRACE ANNOTATOR
108
ORIGINAL SOURCE CODE WITH ANNOTATIONS
110
FILE X' WITH TRACING
FILE Y' WITH TRACING
FILE Z' WITH TRACING
STANDARD COMPILE/LINK/ RUN
104
EXPLICITLY PROGRAMMED OUTPUT
106
STANDARD COMPILE/LINK/ RUN
112
EXPLICITLY PROGRAMMED OUTPUT
114
EXECUTION LOG
116
100
CODE TRACING METHOD

FIGURE 1

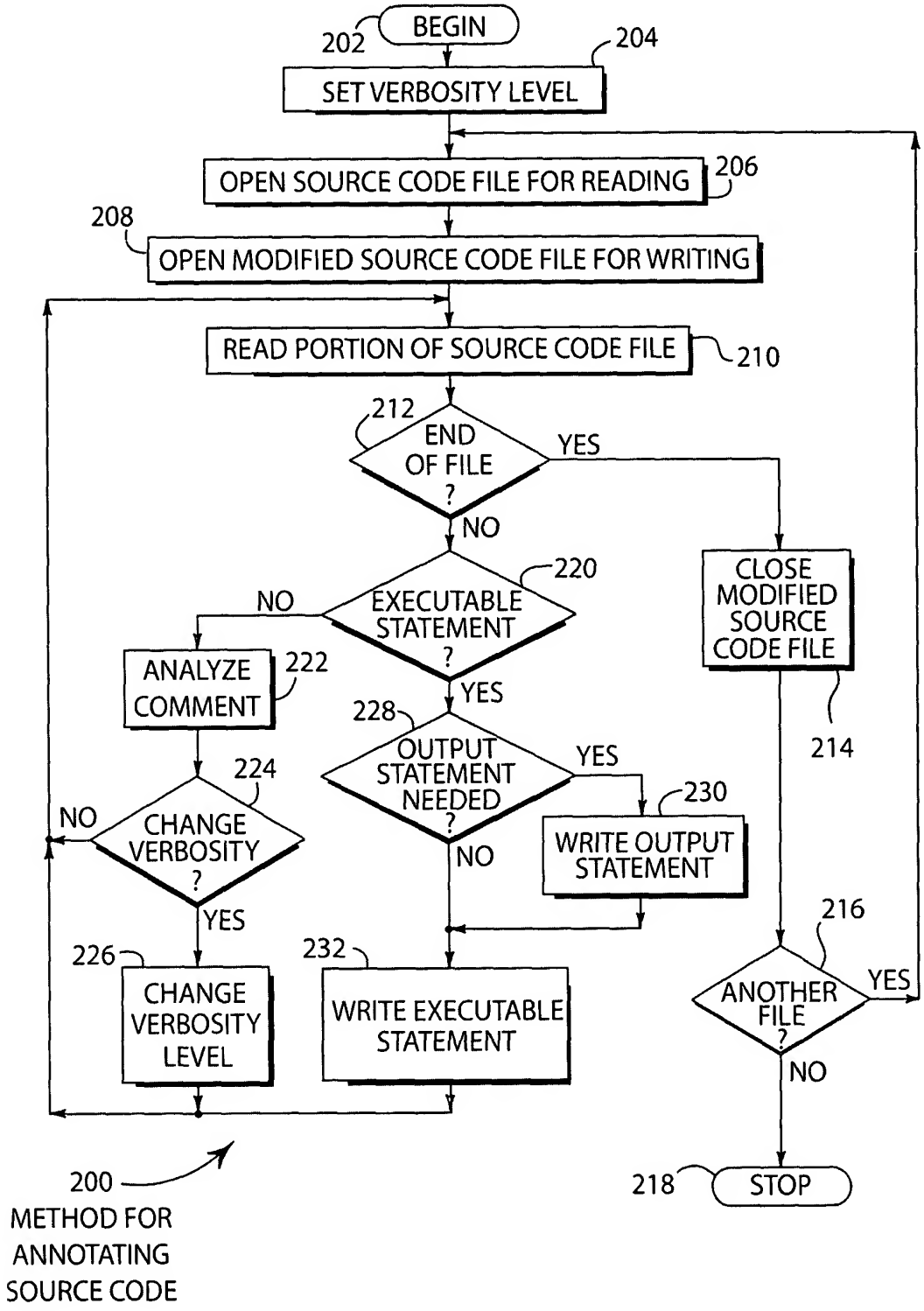
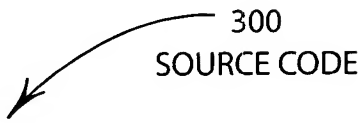

BEGIN
202
204
SET VERBOSITY LEVEL
OPEN SOURCE CODE FILE FOR READING
206
208
OPEN MODIFIED SOURCE CODE FILE FOR WRITING
READ PORTION OF SOURCE CODE FILE
210
212
END OF FILE ?
YES
NO
220
EXECUTABLE STATEMENT ?
NO
YES
CLOSE MODIFIED SOURCE CODE FILE
214
ANALYZE COMMENT
222
228
OUTPUT STATEMENT NEEDED ?
YES
230
WRITE OUTPUT STATEMENT
NO
224
CHANGE VERBOSITY ?
NO
YES
216
ANOTHER FILE ?
YES
NO
226
CHANGE VERBOSITY LEVEL
232
WRITE EXECUTABLE STATEMENT
218
STOP
200
METHOD FOR ANNOTATING SOURCE CODE

FIGURE 2




```
int main( int argc, char *argv[] ) {
   int  inside_comment = 0, trace_declared = 0, trace_opened = 0;
   int i, n;
   char file_name[MAX_LINE_LEN];
   FILE *infile, *outfile;
   char save_dir[MAX_LINE_LEN];

   if(!(argc == 4)) {
      printf("Useage: codetracer <input path> <output path> <file spec>\n\ncodetracer
testdir outdir *.cpp\n" );
      printf("or\n\n\ncodetracer this outdir *.cpp\n" );
      exit( 0 );
   }
   else {
      printf( "Program codetracer version %s\n", VERSION );
   }
   // save path of the directory where we start
   if ( _getcwd( save_dir, _MAX_PATH ) == NULL ) {
      printf( "Could not find path of current directory\n" );
      exit( -2 );
   }
..etc.
```

FIGURE 3

```
FILE *trace;                                    // <<<<< This line is inserted by the tool

int main( int argc, char *argv[] ) {
   trace = fopen( "trace.txt", "wt" ); // <<<<< This line is inserted by the tool
fprintf(trace,"codetracer.cpp 0028:     int main( int argc, char *argv[] ) {\n");
fprintf(trace,"codetracer.cpp 0029:        int  inside_comment = 0, trace_declared = 0,
trace_opened = 0;\n");
   int  inside_comment = 0, trace_declared = 0, trace_opened = 0;
fprintf(trace,"codetracer.cpp 0030:        int i, n;\n");
   int i, n;
fprintf(trace,"codetracer.cpp 0031:        char file_name[MAX_LINE_LEN];\n");
   char file_name[MAX_LINE_LEN];
fprintf(trace,"codetracer.cpp 0032:        FILE *infile, *outfile;\n");
   FILE *infile, *outfile;
fprintf(trace,"codetracer.cpp 0033:        char save_dir[MAX_LINE_LEN];\n");
   char save_dir[MAX_LINE_LEN];

fprintf(trace,"codetracer.cpp 0035:        if(!(argc == 4)) {\n");
   if(!(argc == 4)) {
      printf("Useage: codetracer <input path> <output path> <file spec>\n\ncodetracer
testdir outdir *.cpp\n" );
      printf("or\n\n\ncodetracer this outdir *.cpp\n" );
fprintf(trace,"codetracer.cpp 0038:              exit( 0 );\n");
      exit( 0 );
   }
   else {
fprintf(trace,"codetracer.cpp 0040:        else {\n");
      printf( "Program codetracer version %s\n", VERSION );
   }
fprintf(trace,"codetracer.cpp 0043:        // save path of the directory where we start\n");
   // save path of the directory where we start
fprintf(trace,"codetracer.cpp 0044:        if ( _getcwd( save_dir, _MAX_PATH ) == NULL )
{\n");
   if ( _getcwd( save_dir, _MAX_PATH ) == NULL ) {
      printf( "Could not find path of current directory\n" );
fprintf(trace,"codetracer.cpp 0046:              exit( -2 );\n");
      exit( -2 );
   }   }
..etc.
```

400
ANNOTATED
SOURCE CODE

FIGURE 4

Title: "Automated Code Debugging Tool" Inventor: Grassens Docket No: LSI.75US01 (03-0977)
Law Offices Of William W. Cochran U.S.P.T.O. Customer Number: 27479 Phone: 970.377.6363

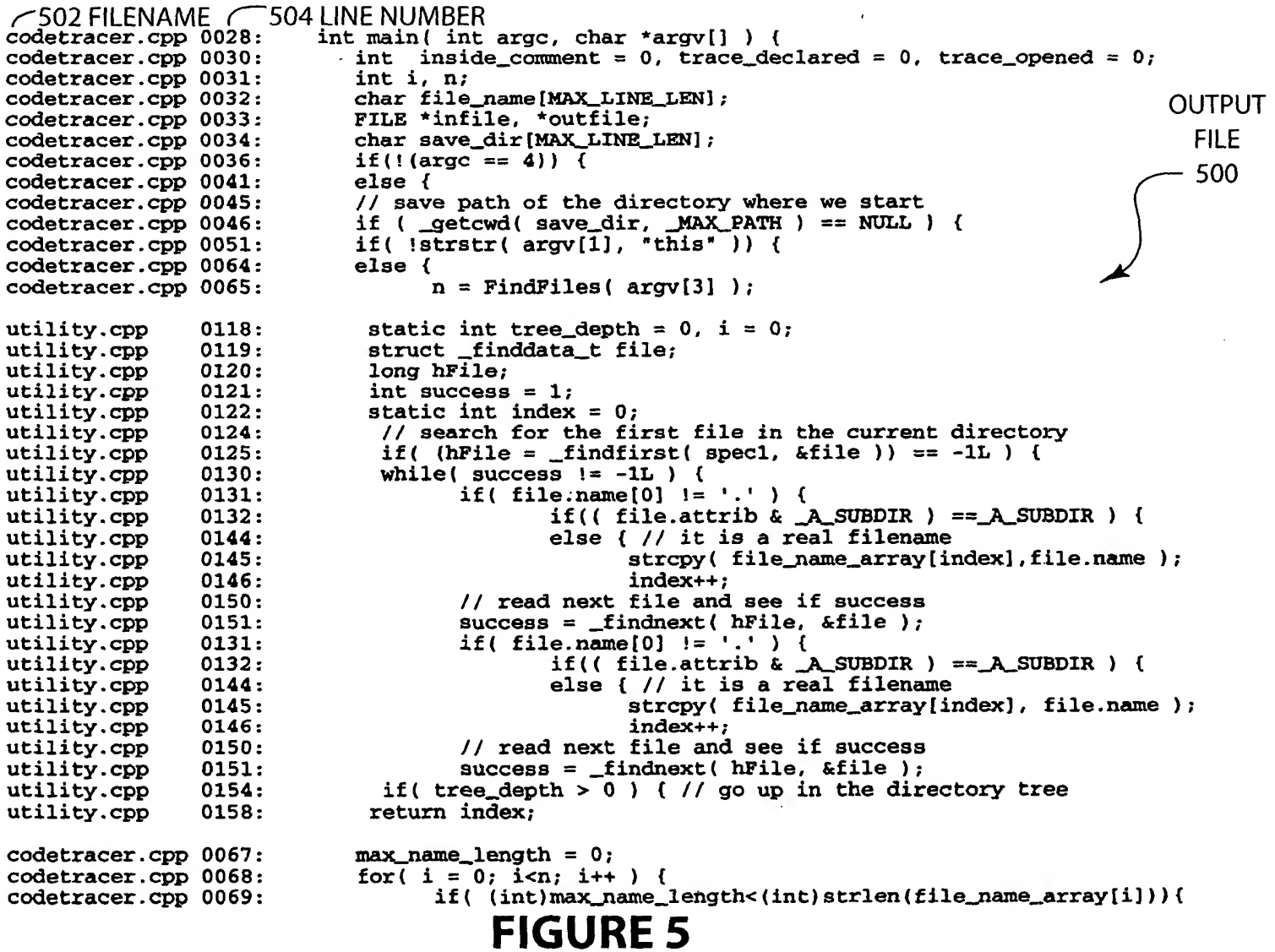


FIGURE 5